PASSED

732A38: Computer lab 4
Computational statistics

Martina Sandberg and Caroline Svahn

March 10, 2016

1 Computations with Metropolis-Hastings

Consider the following probability density function:

$$f(x) \propto x^5 e^{-x}, \quad x < 0$$

You can see that the distribution is known up to some constant of proportionality.

1.1

Use Metropolis-Hastings algorithm to generate samples from this distribution by using proposal distribution as log-normal LN(X_t, 1), take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn-in period, what can be the size of this period?

We want to generate samples from the distribution that is proportional to the probability density function given above, that is some function $\pi(x) = C \cdot x^5 e^{-x}$ where C is some constant. To do this we choose the proposal distribution $q(.|X_t) = LN(X_t, 1)$. To get our sample from the Metropolis-Hastings algorithm we implement a function in `R` that does the following:

1. Initializes the chain $\boldsymbol{X}$ with $X_1 = 1$, here $t = 1$.
2. Generate a candidate point Y from $LN(X_t, 1)$
3. Generate U from a uniform (0,1) distribution
4. If $U \leq \alpha(X_t, Y)$ then set $X_{t+1} = Y$, else set $X_{t+1} = X_t$
5. Set $t = t + 1$ and repeat steps 2 through 5 (until t=10000).

The Metropolis-Hastings algorithm "updates" the last known value by comparing it to the uniform distribution. First, an initial value is set (can be arbitrary). Using the given proposal distribution, a possible update for the initial value is generated. X_t is the current value of X, which in this case means that the deviation of the function which is sampled from is fix, but the mean (which is the function we wish to find) changes as the value of X is updated. A random value from the uniform distribution is then obtained. This is then compared with the equation α. α is defined as:

$$\alpha(X_t, Y) = min\left\{1, \frac{\pi(Y)q(X_t|Y)}{\pi(X_t)q(Y|X_t)}\right\}$$

If the value of $\alpha(X_i, Y)$ is greater or equal to the sample generated from the uniform distribution, the current value of X is updated to the value of Y, otherwise, the current value of X is kept for the next iteration. The procedure is then repeated, either until convergence or for a large number

of iterations.

The result for 10000 iterations is plotted below:

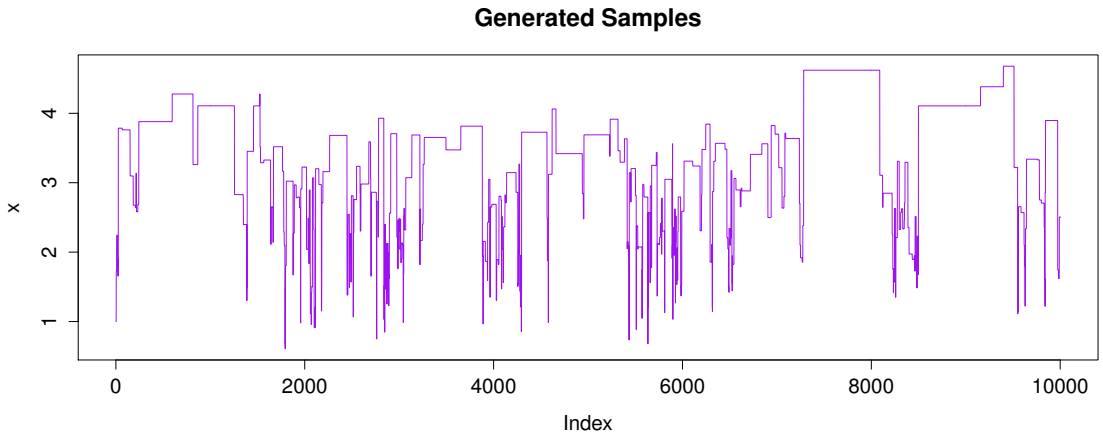

Figure 1: Generated samples from $\pi(x)$ with $LN(X_t, 1)$ as proposal distribution.

The generated samples is plotted in figure 1. The chain is flat most of the time, this is because many points from the proposal distribution is rejected. We can see that the chain is highly irregular under the whole interval and the level is very shifting. Because of this the chain does not converge and thus there is no meaningful burn-in period. The selected proposal density $LN(X_t, 1)$ is not a good one in this case. From testing we can also add that the starting point does matter in this case, so if we were to choose the starting point 6 we would just get a straight line.

1.2

Perform step 1 by using chi-square distribution $\chi^2(floor(X_t + 1))$ as proposal distribution where floor(x) means integer part of x.

Now, instead of using the log-normal distribution, the chi-square distribution with $X_t + 1$ degrees of freedom is used. The result is plotted:

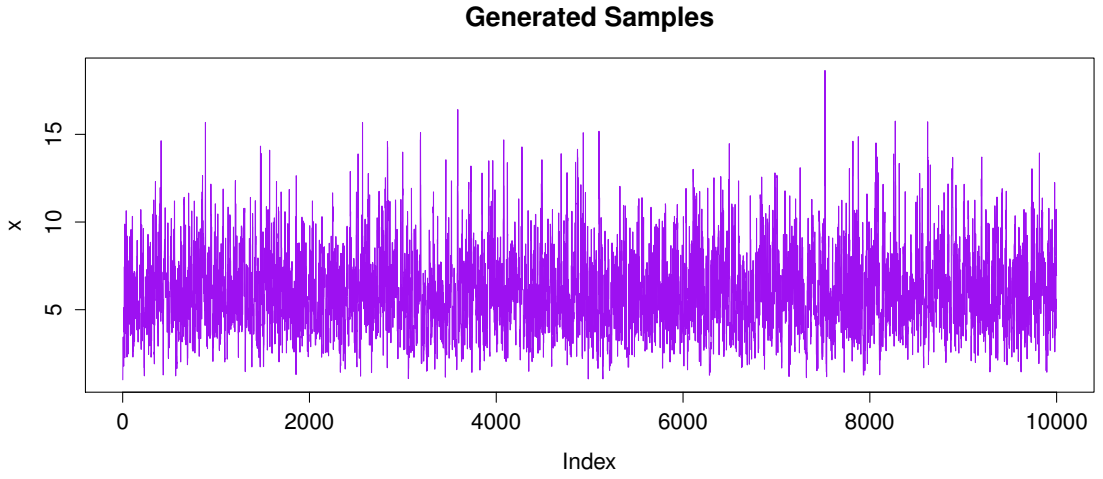

Figure 2: Generated samples from $\pi(x)$ with $\chi^2(floor(X_t + 1))$ as proposal distribution.

Figure 2 shows the generated samples when using the proposal function $\chi^2(floor(X_t+1))$ instead. This chain seems to converge almost directly and therefore the burn-in period is rather short. However, for safety reasons we will use the first 2% as burn-in period. We see that this sample is totally random with a constant mean just like it should be. This proposal distribution seems to be a good one in this case. If we, like we did in the previous example, test with another starting point, like 50, we get the same result. Hence, in this chain, the starting point does not matter.

1.3

Compare the results of steps 1 and 2 and make conclusions.

We can see that when we used the log-normal distribution as proposal distribution the Markov chain did not converge, which it should if we want a reliable result. In the same way we can see that when we used the χ^2-distribution as proposal distribution the Markov chain did converge. The χ^2-distribution seems to be more similar to the target distribution than the log-normal distribution, and therefore the χ^2-distribution is a better proposal distribution.

1.4

Generate 10 MCMC sequences using the generator from the step 2 and with starting points 1,2,. . . , 10. Use Gelman-Rubin method to analyze convergence of these sequences.

The Gelman-Rubin factor is used to check the convergence of multiple MCMC chains run in parallel. IT is calculated as:

$$\sqrt{\hat{R}} = \sqrt{\frac{\hat{var}(v)}{W}}$$

where

$$\widehat{var}(v) = \frac{n-1}{n}W + \frac{1}{n}B$$

where

$$B = \frac{n}{k-1}\sum_{i=1}^{k}(\hat{v}_i - \hat{v})^2$$

and

$$W = \frac{1}{k}\sum_{i=1}^{k}s_i^2$$

Finally,

$$s_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(v_{ij} - \hat{v}_i)^2$$

k is the number of chains and n the number of observations/iterations (after excluding the burn-in). If the Gelman-Rubin factor is about 1 to 1.2, convergence is achieved. In this case, $\sqrt{\hat{R}} = 1.0$, which is indeed within the interval, and one can conclude convergence.

1.5

Estimate $\int_0^\infty x \cdot f(x)dx$ *using the samples from steps 1 and 2.*

We know that for $x_i \epsilon f(x)$ we have that

$$E(x) = \int_0^\infty x f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

The expected value for the random variable x for the sample in step 1 is 3.44 and 6.06 for the sample in step 2 (where we sum over all but the values from the burn-in period).

1.6

The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

$$f(x) = \frac{1}{s^a \Gamma(a)} x^{a-1} exp\left\{-\frac{x}{s}\right\} \tag{1}$$

The Gamma distribution with parameters *shape* $= a$ and *scale* $= s$ has density (1) for $x \geq 0$, $a > 0$ and $s > 0$. The mean is computed as $E(x) = a \cdot s$. We can see that in our case $a = 6$ and $s = 1$. Now we can compute the expected value $E(x) = 6 \cdot 1 = 6$ and we can also conclude that the unknown constant C (as mentioned in part 1) is $C = 1/\Gamma(6) = 1/120$. If we compare the real expected value 6 to the ones obtained in part 5, 3.44 and 6.00, we can conclude that the second approach was more accurate. This is reasonable, since the first attempt did not converge but the second one did.

2 Gibbs sampling

A concentration of a certain chemical was measured in a water sample, and the result was stored in the data **chemical.RData** having the following variables:

- X: day of the measurement
- Y: measured concentration of the chemical.

The instrument used to measure the concentration had certain accuracy; this is why the measurements can be treated as noisy. Your purpose is to restore the expected concentration values.

2.1

Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?

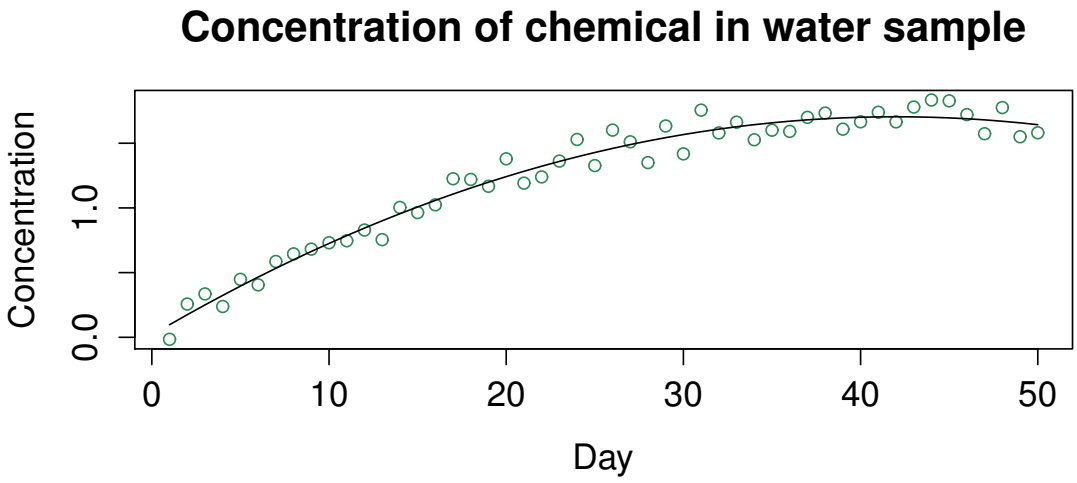

Figure 3: Concentration of a certain chemical in a water sample plotted against the day of measurement.

In figure 3 the dependence is shown above in the plot. A quadratic line is fitted to the observed values, which appears to follow the data nicely. Hence, a model using the quadratic dependence is probably suitable here.

2.2

A researcher has decided to use the following (random-walk) Bayesian model (n=number of observations, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ are unknown parameters):

$$Y_i \sim N(\mu_i, var = 0.2), \;\; i = 1, \ldots, n$$

where the prior is

$$p(\mu_1) = 1$$
$$\mu_{i+1} \sim N(\mu_i, 0.2), \;\; i = 1, \ldots, n-1$$

Present the formulas showing the likelihood $p(\boldsymbol{Y}|\boldsymbol{\mu})$ and the prior $p(\boldsymbol{\mu})$ (hint: a chain rule can be used here $p(\boldsymbol{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)\ldots p(\mu_n|\mu_{n-1}))$

The likelihood of the normal distributed Y_i for $i = 1...n$ is derived as

$$p(\boldsymbol{Y}|\boldsymbol{\mu}) = \prod_{i=1}^{n} f(Y_i; \mu_i) = \prod_{i=1}^{n} C_1 exp\left\{C_2(Y_i - \mu_i)^2\right\} = C_1^n exp\left\{C_2 \sum_{i=1}^{n}(Y_i - \mu_i)^2\right\} \tag{2}$$

The prior - which is also normally distributed - using the formula for the likelihood (only substituting Y_i for μ_i and μ_i for μ_{i-1} since the prior depends on the last value of μ, is as follows:

$$p(\boldsymbol{\mu}) = \prod_{i=2}^{n} f(\mu_i; \mu_{i-1}) = \prod_{i=2}^{n} C_1 exp\left\{C_2(\mu_i - \mu_{i-1})^2\right\} = C_1^{n-1} exp\left\{C_2 \sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2\right\} \tag{3}$$

The likelihood $p(\boldsymbol{Y}|\boldsymbol{\mu})$ is computed as (2) where $C_1 = 1/\sqrt{2\sigma^2\pi}$ and $C_2 = -1/2\sigma^2$. Using the chain rule mentioned in the hint we get the formula for the prior $p(\boldsymbol{\mu})$ to (3).

2.3

Use the Bayes theorem to get the posterior up to a constant of proportionality, and then find out the distributions for $\mu_i|\boldsymbol{\mu}_{-i}, \boldsymbol{Y}$ where $\boldsymbol{\mu}_{-i}$ is a vector containing all μ values except of μ_i.

From Bayes theorem we know that $p(\boldsymbol{\mu}|\boldsymbol{Y}) \propto p(\boldsymbol{Y}|\boldsymbol{\mu})p(\boldsymbol{\mu})$, that is the posterior is proportional to the the likelihood times the prior. Because of the proportionality we can remove all constants. The posterior $p(\boldsymbol{\mu}|\boldsymbol{Y})$ can be computed as (4).

$$\left.\begin{aligned} p(\boldsymbol{\mu}|\boldsymbol{Y}) &\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu_i)^2\right\}\cdot exp\left\{-\frac{1}{2\sigma^2}\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2\right\} \\ &= exp\left\{-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}(Y_i-\mu_i)^2+\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2\right)\right\} \\ &= exp\left\{-\frac{1}{2\sigma^2}\left((Y_1-\mu_1)^2+\sum_{i=2}^{n}(Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2\right)\right\} \end{aligned}\right\} \tag{4}$$

This is then the general form of the posterior. However, the end points will be differently distributed than the i:th observation which means, We are going to get different distributions for $\mu_1|\boldsymbol{\mu}_{-1}, \boldsymbol{Y}$, $\mu_n|\boldsymbol{\mu}_{-n}, \boldsymbol{Y}$ and $\mu_i|\boldsymbol{\mu}_{-i}, \boldsymbol{Y}$ for $i = 2, \ldots, n-1$.. The first and the last μ will be one sum shorter since $\mu_0 = 0$ and there is no μ_{n+1}. Furthermore, the principle of proportionality allows us to exclude constant terms, as before. This means, only terms containing the μ in question will be kept.
For $\mu_1|\boldsymbol{\mu}_{-1}, \boldsymbol{Y}$ we compute as (5) where we can se that we get the normal distribution (6). For $\mu_n|\boldsymbol{\mu}_{-n}, \boldsymbol{Y}$ we compute as (7) where we can se that we get the normal distribution (8). And $\mu_i|\boldsymbol{\mu}_{-i}, \boldsymbol{Y}$ for $i = 2, \ldots, n-1$ computed as (9) which gives the normal distribution (10).

$$\left.\begin{aligned} &\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left((Y_1-\mu_1)^2+(\mu_2-\mu_1)^2\right)\right\} \\ &\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left(-2Y_1\mu_1+\mu_1^2-2\mu_1\mu_2+\mu_1^2\right)\right\} \\ &\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left(2\mu_1^2-2Y_1\mu_1-2\mu_1\mu_2\right)\right\} \\ &\Rightarrow exp\left\{-\frac{1}{\sigma^2}\left(\mu_1^2-Y_1\mu_1-\mu_1\mu_2\right)\right\} \\ &\Rightarrow exp\left\{-\frac{1}{\sigma^2}\left(\left(\mu_1-\frac{Y_1+\mu_2}{2}\right)^2\right)\right\} \end{aligned}\right\} \tag{5}$$

$$\mu_1|\boldsymbol{\mu}_{-1}, \boldsymbol{Y} \sim N\left(\frac{Y_1+\mu_2}{2}, \frac{\sigma^2}{2}\right) \tag{6}$$

$$
\left.
\begin{aligned}
&\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left((Y_n-\mu_n)^2+(\mu_n-\mu_{n-1})^2\right)\right\}\\
&\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left(-2Y_n\mu_n+\mu_n^2-2\mu_{n-1}\mu_n+\mu_n^2\right)\right\}\\
&\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left(2\mu_n^2-2Y_n\mu_n-2\mu_{n-1}\mu_n\right)\right\}\\
&\Rightarrow exp\left\{-\frac{1}{\sigma^2}\left(\mu_n^2-Y_n\mu_n-\mu_{n-1}\mu_n\right)\right\}\\
&\Rightarrow exp\left\{-\frac{1}{\sigma^2}\left(\left(\mu_n-\frac{Y_n+\mu_{n-1}}{2}\right)^2\right)\right\}
\end{aligned}
\right\} \tag{7}
$$

$$
\mu_n|\boldsymbol{\mu}_{-n},\boldsymbol{Y}\sim N\left(\frac{Y_n+\mu_{n-1}}{2},\frac{\sigma^2}{2}\right) \tag{8}
$$

$$
\left.
\begin{aligned}
&\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left((Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2\right)\right\}\\
&\Rightarrow exp\left\{-\frac{1}{2\sigma^2}\left(-2Y_i\mu_i+3\mu_i^2-2\mu_i\mu_{i-1}-2\mu_{i+1}\mu_i\right)\right\}\\
&\Rightarrow exp\left\{-\frac{3}{2\sigma^2}\left(-\frac{2}{3}Y_i\mu_i+\mu_i^2-\frac{2}{3}\mu_i\mu_{i-1}-\frac{2}{3}\mu_{i+1}\mu_i\right)\right\}\\
&\Rightarrow exp\left\{-\frac{3}{2\sigma^2}\left(\left(\mu_i-\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3}\right)^2\right)\right\}
\end{aligned}
\right\} \tag{9}
$$

$$
\mu_i|\boldsymbol{\mu}_{-i},\boldsymbol{Y}\sim N\left(\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3},\frac{\sigma^2}{3}\right) \tag{10}
$$

2.4

Use the distributions derived in step 3 to implement a Gibbs sampler that uses $\boldsymbol{\mu^0}=(0,\ldots,0)$ *as a starting point. Run the Gibbs sampler to obtain 1000 values of* $\boldsymbol{\mu}$ *and then compute the expected value of* $\boldsymbol{\mu}$ *by using Monte Carlo approach. Plot the expected value of* $\boldsymbol{\mu}$ *versus X and Y versus X in the same graph. Does it seem that you have managed to remove the noise? Does it seem that the expected value of* $\boldsymbol{\mu}$ *can catch the true underlying dependence between Y and X?*

Gibbs sampling uses previous estimations to update the value of μ, as derived in the formulas above. In short terms, the updates are done a thousand times for every point estimate of μ, using the last known updates of the μ_i:s desired for the formula of derivation. When all values are obtained, the expected values of μ is obtained as the mean of each column in the matrix obtained when iterating (meaning, the mean of each point estimate $\mu_{,i}$).

The plot below shows the result:

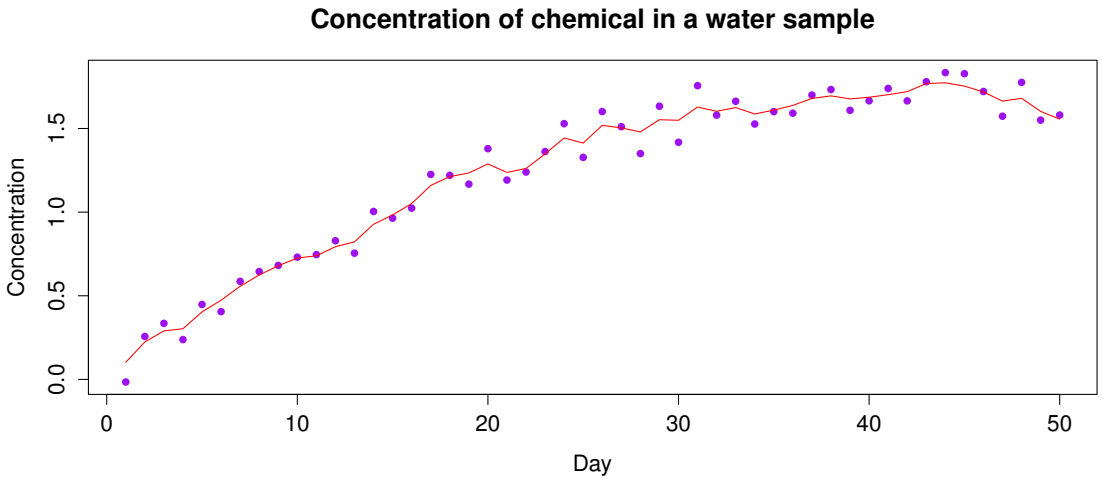

Figure 4: Concentration of a certain chemical in a water sample (purple dots) and expected values of the concentration (red line) plotted against the day of measurement.

In figure 4 the expected μ-values (the red line) and Y (the purple dots) are plotted against X. We can see that these expected μ-values have a lower variance so we have managed to reduce the noise. It seems that the expected value of $\boldsymbol{\mu}$ can catch the true underlying dependence between Y and X.

2.5

Make a trace plot for μ_{50} and comment on the burn-in period and convergence.

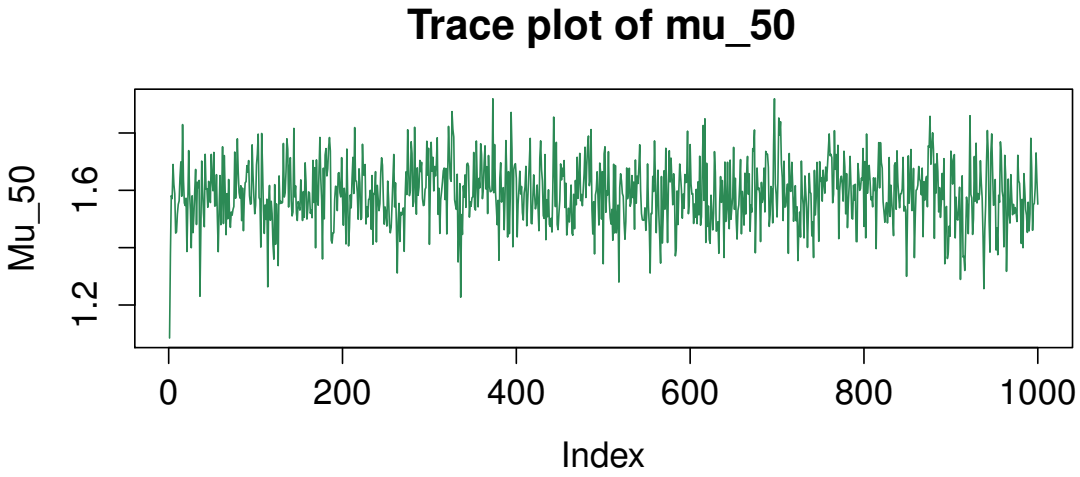

Figure 5: Trace plot for the expected value of Y.

In figure 5 a trace plot for μ_{50} is shown. This chain seems to have a constant mean so it does converge. Its hard to say exactly how long the burn in period is because it seems to converge quite fast, so 1-2% might be reasonable.
There is however a jump around $j = 250$. The optimal choice here might then be to repeat the process for more values and use the first 500 values as a burn-in.

Appendix

Contribution

The text (algorithm descriptions and results/interpretations of results) is merged from both reports - sometimes, one report is clarifying or giving the explanations in easier or more scientific words. The plots are from both reports, since the results are the same. Some of the equations are from Martina's report and some from Caroline's (we both have most of the equations included in the individual reports). The code is from Caroline's report but Martina's is basically the same and yields, on the whole, the same results.

R-code

```
#1.1
a<-2000
Y<-numeric(a)
X<-numeric(a)

X[1]<-1

set.seed(12345)
for (t in 1:a) {
  Y[t]<-rlnorm(n=1,meanlog=X[t])
  U<-runif(1)

  ta<-(Y[t]^5*exp(-Y[t]))*dlnorm(X[t],meanlog=Y[t])
  na<-(X[t]^5*exp(-X[t]))*dlnorm(Y[t],meanlog=X[t])
  frac<-ta/na
  alpha<-min(1,frac)

  if (U<=alpha) {
    X[t+1]<-Y[t]
  } else {
    X[t+1]<-X[t]
  }
}

plot(X, type="l", col="seagreen", xlab="Iteration",
       main="Metropolis-Hastings chain")

#1.2

a<-2000
Yc<-numeric(a)
Xc<-numeric(a)

Xc[1]<-1

for (t in 1:a) {
  Yc[t]<-rchisq(n=1,df=floor(Xc[t]+1))
  U<-runif(1)

  ta<-(Yc[t]^5*exp(-Yc[t]))*dchisq(Xc[t],df=floor(Yc[t]+1))
  na<-(Xc[t]^5*exp(-Xc[t]))*dchisq(Yc[t],df=floor(Xc[t]+1))
  frac<-ta/na
  alpha<-min(1,frac)
```

```
  if (U<=alpha) {
    Xc[t+1]<-Yc[t]
  } else {
    Xc[t+1]<-Xc[t]
  }
}

plot(Xc, type="l", col="seagreen",xlab="Iteration", ylab="X",
       main="Metropolis-Hastings chain")

#1.4
s<-numeric(10)
mean<-numeric(10)

for (i in 1:10) {
  a<-2000
  Yc<-numeric(a)
  Xc<-numeric(a)

  Xc[1]<-i

  for (t in 1:a) {
    Yc[t]<-rchisq(n=1,df=floor(Xc[t]+1))
    U<-runif(1)

    ta<-(Yc[t]^5*exp(-Yc[t]))*dchisq(Xc[t],df=floor(Yc[t]+1))
    na<-(Xc[t]^5*exp(-Xc[t]))*dchisq(Yc[t],df=floor(Xc[t]+1))
    frac<-ta/na
    alpha<-min(1,frac)

    if (U<=alpha) {
      Xc[t+1]<-Yc[t]
    } else {
      Xc[t+1]<-Xc[t]
    }
  }
  s[i]<-sd(Xc[201:2000])
  mean[i]<-mean(Xc[201:2000])
  plot(Xc, type="l", main=i)
}

W<-sum(s)/10
B<-(1800/9)*sum((mean-mean(mean))^2)

overall<-((1800-1)/1800)*W+(1/1800)*B
gelRub<-sqrt(overall/W)
#1.0027 --> convergence

#1.5
int1<-mean(X[201:2000])
#3.61

int2<-mean(Xc[201:2000])
#5.959904

#1.6
#beta=1, alpha=6
```

```
#E(X)=alpha*beta=6*1=6


#2.1
plot(X,Y, col="seagreen",xlab="Day", ylab="Concentration",
     main="Concentration of chemical in water sample")
lines(predict(lm(Y~X+I(X^2))))
#quadratic is suitable

#2.4

set.seed(12345)
n<-length(X)
mu<-matrix(0,nrow=1000,ncol=n)

for (j in 1:1000) {
  for (i in 1:n) {
    if (i==1) {
      if (j==1) {
        mu[j,1]<-rnorm(1,mean=((Y[1]+mu[j,2])/2),sd=(0.2/2))
      } else {
        mu[j,1]<-rnorm(1,mean=((Y[1]+mu[j-1,2])/2),sd=(0.2/2))
      }
    } else if (i==n) {
      mu[j,n]<-rnorm(1,mean=((Y[n]+mu[j,n-1])/2),sd=(0.2/2))
    } else {
      if (j==1) {
        mu[j,i]<-rnorm(1,mean=((Y[i]+mu[j,i-1])/3),sd=(0.2/3))
      } else {
        mu[j,i]<-rnorm(1,mean=((Y[i]+mu[j,i-1]+mu[j-1,i+1])/3),sd=(0.2/3))
      }
    }
  }
}

muVec<-apply(mu, 2, mean)
plot(X,Y, col="seagreen",xlab="Day", ylab="Concentration",
     main="Concentration of chemical in water sample")

lines(muVec)

#2.5
plot(mu[,50], type="l", col="seagreen",ylab="Mu_50",
     main="Trace plot of mu_50")
```